

wCorr Arguments

Paul Bailey, Ahmad Emad, (people who do QC for this product)

2016-03-11

This vignette explores two Boolean switches in the wCorr package. First, the `ML` switch allows for either a non-MLE (but consistent) estimate of the nuisance parameters that define the binning process to be used (`ML=FALSE`) or for the nuisance parameters to be estimated using the MLE (`ML=TRUE`). Second the `fast` argument gives the option to use a pure R implementation (`fast=FALSE`) or an implementation that relies on the `Rcpp` and `RcppArmadillo` packages (`fast=TRUE`).

The *wCorr Formulas* vignette describes the statistical properties of the correlation estimators in the package and has a more complete derivation of the likelihood functions.

The ML switch

The correlation coefficients between two vectors of random variables that are jointly bivariate normal—call the vectors \mathbf{X} and \mathbf{Y} .

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N \left[\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \Sigma \right]$$

where $N(\mathbf{A}, \Sigma)$ is the bivariate normal distribution with mean \mathbf{A} and covariance Σ .

Polyserial computation

the likelihood function for an individual observation of the polyserial is¹

$$\Pr(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i) = \phi(z_i) \left[\Phi \left(\frac{\theta_{m_i+2} - r \cdot z_i}{\sqrt{1 - r^2}} \right) - \Phi \left(\frac{\theta_{m_i+1} - r \cdot z_i}{\sqrt{1 - r^2}} \right) \right]$$

where ρ is the correlation between \mathbf{X} and \mathbf{Y} , \mathbf{Z} is the normalized version of \mathbf{X} , and \mathbf{M} is a discretized version of \mathbf{Y} , using $\boldsymbol{\theta}$ as cut points as described in the *wCorr Formulas* vignette.

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}; z, m) = \sum_i w_i \ln [\Pr(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i)]$$

The derivatives of ℓ can be written down but are not readily computed and so when the `ML` argument is set to `FALSE` (the default) a one dimensional optimization of ρ is calculated using `stats::optimize`. When the `ML` argument is set to `TRUE` a multi-dimensional optimization is done for ρ and $\boldsymbol{\theta}$ using `minqa::bobyqa`.

¹See the *wCorr Formulas* vignette for a more complete description and motivation for the polyserial correlations's likelihood function.

Polychoric computation

For the polychoric the observed data is discreteized for both variables. Here the discretized version of \mathbf{X} is \mathbf{P} and the discretized version of \mathbf{Y} remains \mathbf{M} .² The likelihood function for the polychoric is

$$\Pr(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i) = \int_{\theta'_{p_i+1}}^{\theta'_{p_i+2}} dx \int_{\theta_{m_i+1}}^{\theta_{m_i+2}} dy f(x, y | \rho = r)$$

where $f(x, y | r)$ is the normalized bivariate normal distribution with correlation ρ , and $\boldsymbol{\theta}'$ are the cut points used to discretize \mathbf{X} into \mathbf{P} .

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}, \boldsymbol{\theta}'; \mathbf{p}, \mathbf{m}) = \sum_i w_i \ln [\Pr(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i)]$$

The derivatives of ℓ can be written down but are not readily computed and so when the ML argument is set to **FALSE** (the default) a one dimensional optimization of ρ is calculated using `stats::optimize`. When the ML argument is set to **TRUE** a multi-dimensional optimization is done for ρ , $\boldsymbol{\theta}$, and $\boldsymbol{\theta}'$ using `minqa::bobyqa`.

General setup for the unweighted case

A simulation is run several times. For each iteration, the following procedure is used:

- select the number of observations (n)
- select a true correlation coefficient ρ
- generate \mathbf{X} and \mathbf{Y} to be bivariate normally distributed using a pseudo-Random Number Generator (RNG)
- using a pseudo-RNG, select the the number of bins for \mathbf{M} and \mathbf{P} (t and t') independantly from the set $\{2, 3, 4, 5\}$
- select the bin boundaries for \mathbf{M} and \mathbf{P} ($\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$) by sorting the results of $(t - 1)$ and $(t' - 1)$ draws, respectively, from a normal distribution using a pseudo-RNG
- confirm that at least 2 levels of each of \mathbf{M} and \mathbf{P} are occupied (if not, rerun to generating \mathbf{X} and \mathbf{Y})
- calculate and record relevant statistics

When the exact method of selecting a parameter (such as n) is not noted in the above description it is described as part of each simulation.

ML switch

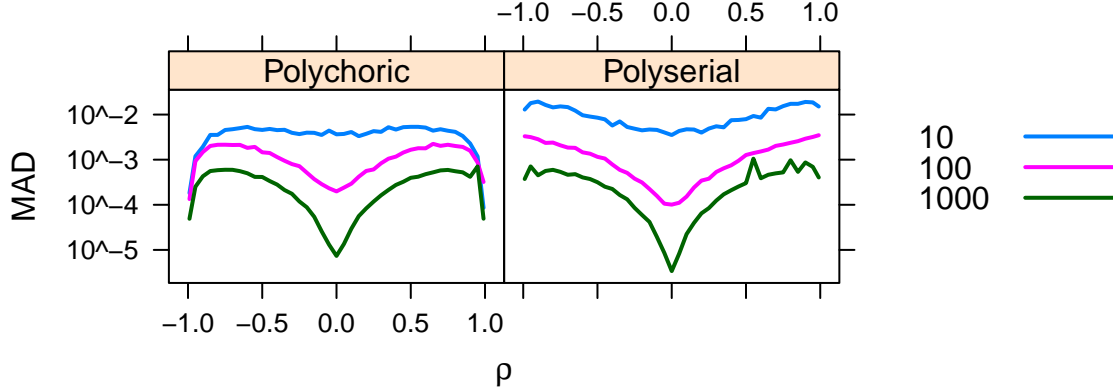
A simulation was done at each level of the cartesian product of $\text{ML} \in \{\text{TRUE}, \text{FALSE}\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, \dots, 0.95, 0.99)$, and $n \in \{10, 100, 1000\}$. For precision, each iteration is run three times. The computation is run so that the same values of the variables are used for $\text{ML}=\text{TRUE}$ as $\text{ML}=\text{FALSE}$ and then the statistics are compared between the two sets of results. where MAD is the mean absolute difference and is given by

$$MAD = |r_{ML=TRUE} - r_{ML=FALSE}|$$

²See the “wCorr Formulas” vignette for a more complete description and motivation for the polychoric correlations’s likelihood function.

where $r_{ML=TRUE}$ is the estimated correlation when $ML=TRUE$ and $r_{ML=FALSE}$ is the estimated correlation when $ML=FALSE$.

This is a plot of the MAD as a function of the true correlation coefficient. It shows a decrease in MAD as n increases (change from line to line), a decrease when the correlations are in the neighborhood of zero that is more pronounced for larger n and a dip then the correlation is exactly 1 or -1.



This table shows the MAD by n and correlation type.

Correlation type	n	MAD
Polychoric	10	0.0038823
Polychoric	100	0.0012159
Polychoric	1000	0.0003116
Polyserial	10	0.0098884
Polyserial	100	0.0013830
Polyserial	1000	0.0003397

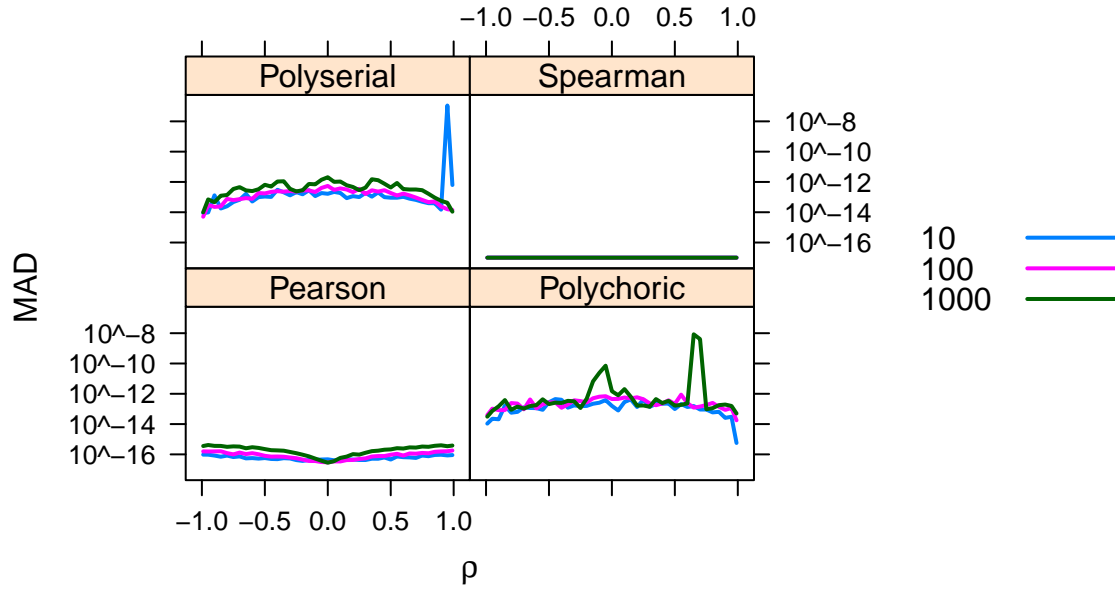
These results show that the side of the MAD decreases as n increases. When $n = 10$, the MAD is less than 0.02 for the polyserial and 0.002 for the polychoric.

fast switch

This section looks at the agreement between the pure R implementation of the optimizations and the `Rcpp` and `RcppArmadillo` implementation. The code can compute with either option by setting `fast=FALSE` (pure R) or `fast=TRUE` (`Rcpp`).

A simulation was done at each level of the cartesian product of $\text{fast} \in \{TRUE, FALSE\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, \dots, 0.95, 0.99)$, and $n \in \{10, 100, 1000\}$. Each iteration was run 100 times. The computation is run so that the same values of the variables are used for `fast=TRUE` as `fast=FALSE` and then the statistics are compared between the two sets of results.

This is the summary of all differences between the `fast=TRUE` and `fast=FALSE` runs for the polyserial. Note that differences smaller than 10^{-16} are indistinguishable from 0 by the machine. However, a factor of 10^{-17} was added to the results so that they could all be shown on a log scale. Thus the Spearman never shows differences that are different from zero.

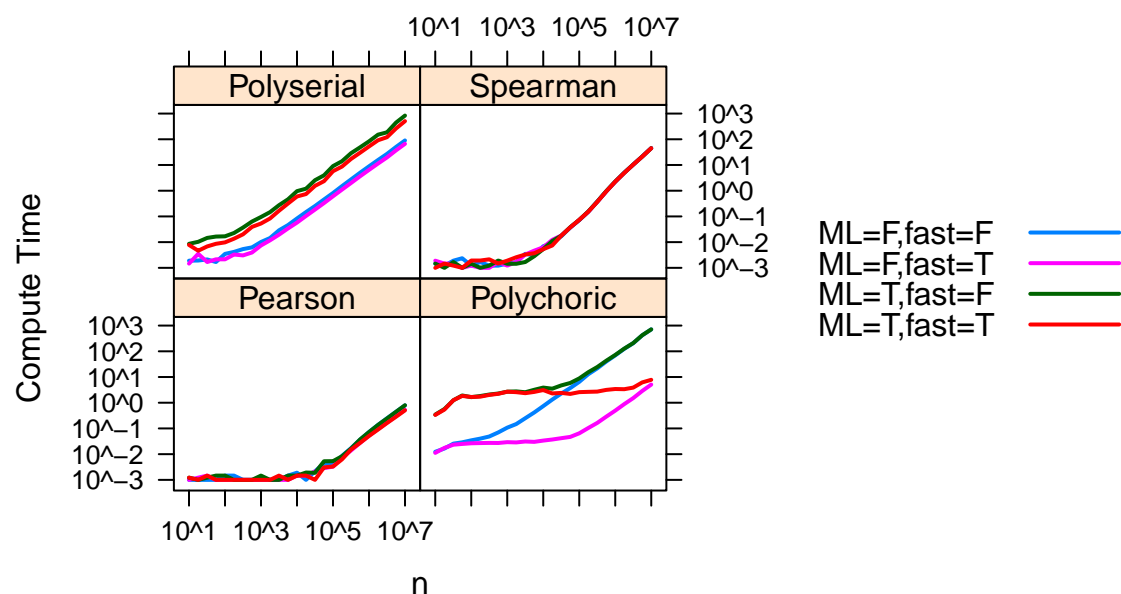


The above shows that differences as a result of the **fast** argument are never larger than 10^{-8} for any type.

Implications for speed

A simulation was done at each level of the cartesian product of $ML \in \{\text{TRUE}, \text{FALSE}\}$, $\text{fast} \in \{\text{TRUE}, \text{FALSE}\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, \dots, 0.95, 0.99)$, and $n \in \{10^1, 10^{1.25}, 10^{1.5}, \dots, 10^7\}$. For precision, each iteration is run 80 times when $n < 10^5$ and 20 times when $n \geq 10^5$. The computation is run so that the same values of the variables are used all four levels of **ML** and **fast**. The variety of correlations is chosen so that the results represent an average of possible values of ρ .

The following plot shows the mean compute time versus n .



Conclusion

Using tables presented in this vignette, users who wish to use the more accurate `ML=TRUE` argument can compare the difference in computation time and the difference in results.

the `fast` argument is provided primarily for comparison of the `Rcpp` and pure R code and shows agreement to within 10^{-8} .